**EVOLVING LANDSCAPE OF WEB APPLICATION SECURITY: FROM COMMON THREATS TO EMERGING CHALLENGES**

# Abstract

A critical aspect of modern web development is web application security, which ensures that websites and online services are safeguarded against a variety of cyber threats and vulnerabilities. Given that web applications handle sensitive data, such as personal and financial information, they often become prime targets for malicious attacks. This section provides an overview of common security threats faced by web applications, including SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), broken authentication, and sensitive data exposure. It also highlights best practices for mitigating these risks, such as input validation, implementing strong authentication and access controls, encrypting data, managing secure sessions, and performing regular software updates.By adopting these security measures, organizations can protect user data, prevent unauthorized access, and ensure the integrity and availability of their web applications. It's important to recognize that web security is an ongoing process, necessitating continuous monitoring, testing, and updates to defend against evolving threats. This makes it a fundamental responsibility for both developers and organizations. Keywords: web applications, threats, attacks, security

# Overview of Web Application Security

The whole programming depends on the web security features, which defend online services from various security threats. Application security is the most essential phase of any website platform. It helps to implements protect some good and protective firewalls, security features, errors, and malware functions that exploit code, configuration, or design vulnerabilities. Since web applications are increasingly used to handle sensitive data, such as personal information, and other confidential information, that ensures their security has become paramount [1]. Web application security involves both protection of the application itself and the data transmitted between the server and users. These projects introduce some essential server attacks of SQL injections like SQLlite Injection SQLi, Cross-Site Scripting XSS, Broken Authentication, Cross-Site Request Forgery CSRF, Sensitive Data Exposure, etc.

1:SQL Injection Attack On Database: Web security is a part of protecting web servers and sites from various malfunctions attacks. The most common attacks in SQL platforms are injection attacks and some high-profile data breaches in the server system of the web platforms. The attacker injects malicious code into input fields such as the login forms to generate some unauthorized attack on the main frame.

1. Cross-Site Scripting On XSS: These attacks directly involve injecting malicious scripting in the web platforms. They execute the browser experience and impact of the user who visited those web pages.
2. Broken Authentication: This attack is the main phase when the hackers can crack passwords to ensure successful data breaches from the mainframes of the server [2]. The attacker sends some cookies and unauthorized spam sms to enter the main frame.
3. Sensitive Data Exposure: Many websites are performed on the web platforms and many times the attacker wants to crash some sensitive data like financial data, health information, and personal information.

# Methodology for Vulnerability Classification

Web security is plagued by various vulnerabilities, including Cross-Site Scripting (XSS), broken authentication, and sensitive data exposure. These issues can lead to data breaches and unauthorized access to compromised systems.

**Cross-Site Scripting (XSS)**

Cross-Site Scripting (XSS) is one of the most prevalent vulnerabilities in web applications. In an XSS attack, an attacker injects malicious scripts into a website, which are then executed in the browsers of unsuspecting users. This process can compromise sensitive data, including cookies, session tokens, and other information accessible through the browser.

XSS attacks can be classified into three main types:

**Stored XSS**: In this type of attack, the malicious script is permanently stored on the target server, such as in a database. The script is retrieved and executed whenever a user accesses the affected pages, potentially leading to the unauthorized sharing of personal or sensitive information.

**Reflected XSS**: Here, the malicious script is injected into a web server and is often executed via a search query or input field. Attackers may trick users into clicking a specially crafted URL that activates the script.

**DOM-based XSS**: In this variant, client-side scripts manipulate the Document Object Model (DOM) without proper validation, allowing the execution of injected scripts.

**Application Security Testing**

Regular testing of web applications for vulnerabilities is essential. This can be achieved through automated tools such as vulnerability scanners, penetration testing by security experts, and secure code reviews. Such testing helps identify and resolve potential vulnerabilities before attackers can exploit them.

# Detailed Analysis of Common Vulnerabilities (e.g., Injection, Broken Authentication, XSS, etc.)

Web applications security systems are projects that identify the most critical web scripting security risk, serving as a foundation for understanding some common enamelware attacks and common threats and vulnerabilities. One such implementation is:

*“Sample PHP (the wrong way) $recipient = $\_POST[‘recipient’];*

*$sql = "SELECT PersonID FROM Person WHERE Username='$recipient'";*

*$rs = $db->executeQuery($sql); P bl ro em What if ‘recipient’ is a malicious string that changes the meaning of the query”*

The most common vulnerabilities are

1: SQL Injection (SQLI)

2: Cross Site Scripting

3: Broken Authentication

**1 SQL Injections on common Vulnerabilities:**

In the most prevalent of web application vulnerabilities SQL injection remains one of them. The injector is allowed to insert an attacker to identify them and block their access. In this attack, the attacker wants to crash some data bridges, block some user interfaces, and try to drop down server performance [6]. The system software allows the complete disclosure of all data on the system. t either corrupts the data or renders it inaccessible, effectively taking control of the database server. Subsequently, the attackers leverage malware functions and viruses within the SQL command. After the injection process, the attacker wants to send unauthorized data to interrupt the server connection, (server communication, for example: blocking GET and FETCH requests) to access or manipulate data sets. According to the web application security projects, the risk factor is generated to get some extra security benefit.

* Attack mechanism

The cyber attackers send some unauthorized input seedlings that can modify the SQL format structure to directly affect the server systems. In the mainframes, the attacker continuously attacks the main server system to destroy files [7]. The main aim of server attacks is to destroy the server communication, cross data, and try to destroy files. n this scenario, the data is encrypted, and the attack targets this encrypted format to uncover the actual resource [8]. Cybercriminals have various motivations to create cyber attacks. Some attacks are targeting personal systems to access a company’s financial reports. In the main data frame of the server site, the data are extracted sensitively in the database or even, deleted records from the database using some dropdown command like DROP TABLE *table\_name*;

* Prevention Techniques

“$query = sprint f("SELECT FROM WHERE name='%s'",

$mysqli->real\_escape\_string($city));

$result = $mysqli->query($query);

Most modern techniques are required to implement some security buffers in modern website applications connected to databases programmed using Structured Query Language (SQL). SQL injection (SQLi). Vulnerabilities arise when websites do not adequately screen, filter, or control the queries from the website, which permits attackers to attempt to inject fragments of SQL code into database queries to extract information [9]. To prevent certain unusual queries and data requests, SQL attacks are carried out on web platforms to exploit functional errors. The main 5 stages are specially required to generate some essential methods to get regular output like:

1: Filter Database Inputs: In these terms, the data are required to prevent cybercriminals. The attackers continuously attack the mainframe of the server system. At this time the data is corrupted and makes some essential errors to access the mainframe. The final stage of this part is filtering and deleting some unauthorized code implementations from the user input [10].

2: Restricted Database code: The main server is run to implement and generate a smooth process of quarry performance that reflects in the user experiences. Some restrictions are generated in the mainframe of system software that creates the actual problems. To prevent exploration by limiting database procedures and coding some unintended database queries to get an exact goal.

3: Restricted database Access: he next steps involve data access and a restricted data process, which are the primary methods for ensuring effective data protection. It helps to identify unauthorized access and block them instantly.

4: Maintaining database and Applications: the next step involves regularly maintaining data sets, following the patching of your data frames and server room to ensure everything remains editable and up to date whenever possible.

5: Monitoring database input and communications [11]: The most essential and effective step in the mainframe of any server hum is monitoring some nonessential threads and viruses that can be very harmful to server systems. The server monitoring helps miscellaneous SQL attack attempts.

**2 Cross-site scripting (XSS):**

The main work frames depend on the server side in the XSS vulnerabilities are related when the applications include untrusted data frames in the web pages without proper scanning and validations [12]. This is the main server fetching technique to enable attackers to inject some unauthorized spam and miscellaneous scripting that can steal session cookies to implement data blockades on behalf of user interference [13]. “ This report considers various forms of XSS like Stored XSS, Reflected XSS, and the last DOM-based XSS. These are the major types of XSS attacks. “

* Types of XSS attacks

1: Stored XSS: The effective and essential part of any web scripting is the XSS site scripting format [14]. The unused and miscellaneous data scripting is permanently stored in the regular update and interferences to the targeted server in the database. The database is getting some conscious results and is executed whenever a user visits the site pages of the affected side.

2: Reflected-based XSS: This is another effective way to attack a web database, the actual problem is generated when the users click on functions like links or spam tabs. Calculating the server to reflect the scripting information to the user interface [15].

3: DOM-based XSS: In this attack, the user raised a ticket as in the link frame, and exploited client-to-user side scripts that “intercepted some data processing to achieve maximum strength [16].

“Here are some examples” <? Php

$name = $\_GET ['name'];

Echo "Welcome $name<br>";

Echo "<a href="http://xssattackexamples.com/">Click to Download</a>";?>

index.php? Name=guest<script>alert ('attacked') </script> (the attack URL)”

* Preventions technique

Cross-site prevention is a crucial element in safeguarding against vulnerabilities in server interactions. To prevent some cross-side XSS, web platforms should be authorized to sanitize user inputs and user threads to improve the potential strength of networking systems [17]. XSS prevention techniques are vital for enforcing security policies that protect against common threats in the user interface. These techniques include blacklist filtering, whitelist filtering, contextual encoding, input validation, and implementing a Content Security Policy

**3 Broken Authentications:**

Broken Authentication is the most important phase of web scripting platforms to prevent some essential web scripting techniques. You must create some new accounts and handle some essential parts of any web application [18]. These are the features the system creates for the user login system to log in and access their actual targets. Weakness in authentication allows an attacker to compromise a user interface. The common and regular issue during this phase is weak password strength, the attacker delivers some salt scripting process in the main password to extract and crack it [19]. The attack targets the main base of server systems to get source downs. Multi-factor authentication is one of them to directly impact any server systems and the management session flaws. Broken authentications lead to account hacking and spam access to sensitive data prevention.

* Risk factor

This is the main factor of any web security management environment that can manage the exceptional risk factors of any web server systems [20]. Risk management is also called a defence system of any server system. In the web application, there are some crucial and essential risk factors are introduced:

1. Weak password policy: the user allows the user to create some weak and less complex passwords and they increase the risk of the brute force attacking technique.
2. Session fixation attacks: as the data fixation attack the attackers want to exploit a user phase session ID to take control after the user has authentication.

* Preventions techniques

The business has generated some strong system security management to improve strong password policies and influence the enforcement of multi-factor authentication. Regularly reviewing session management policies is essential to ensure the effectiveness of the logout function and its impact on user sessions [21].

**4 Cross-site Request forgery**

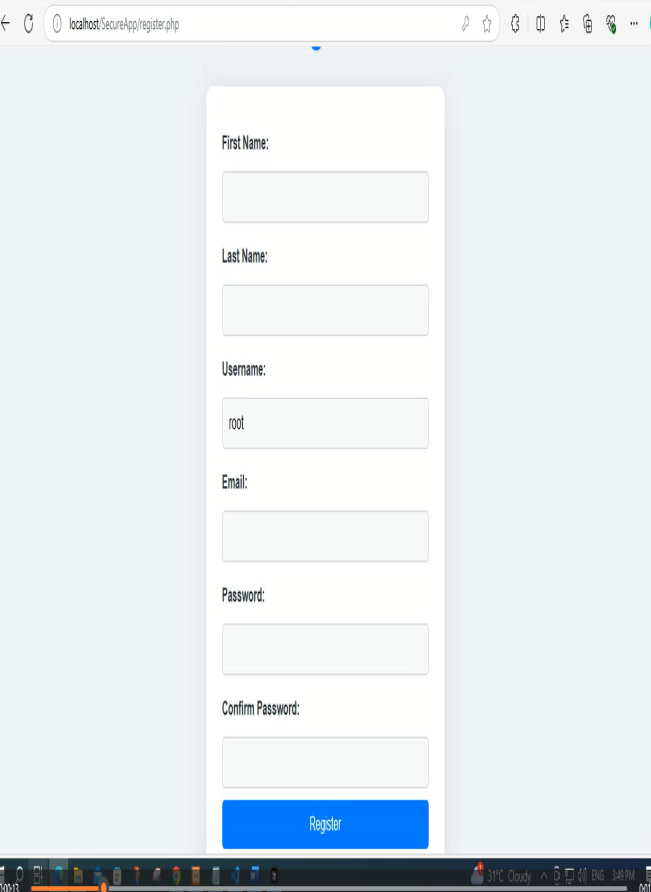
CSRF is the main frame in the web platforms that exploits the trust that a web platform has in the user website. The CSRF is also known as Cross-Site Request Forgery (XSRF), the site is required to generate some effective threads to implement some security implementations [22]. In this type of attack, the attacker sends some random link and email. A successful security bridge attack involves affecting the user and business website performances.

* Attacking mechanism

In the CSRF attack performed The attackers send a get request in the web applications on behalf of any user authentications. After the site crash, the attackers want to embed miscellaneous coding salt in the email and the web server [23]. The users are logged in to the applications process that request without any future verifications process. Leading to some unauthorized action in the mainframe of the database.

* Prevention technique

Protecting against CSRF attacks is essential for developers to implement effective security measures. The developers can generate some exceptional implementations of anti-CSRF tokens that are very much unique to each user [24]. These tokens are generated to create state-changing requests and ensure the actual legitimate.

****

**Figure: WebHost**

(Source: Self-created)

# Emerging Vulnerabilities (e.g., API-related vulnerabilities)

**1 API Related Vulnerabilities:**

The increasing usage of API has introduced new challenges for security functions. The applications run into the server site protocol infrastructure and generate some business classifications to improve security strength. Vulnerabilities such as incorrect authentications to improve signal strength. The main course related to this field correctly covers the lack of encryption, some prevalent interfaces, and insufficient rate of limiting [25]. The APIS is often designed to implement less security challenges methods in web applications. After that, the data makes some attractive targets for the attacker's site.

**Common API Vulnerabilities**:

1:Insecure Endpoints implementations: API may expose some sensitive data information without proper accessibility control [26].

**2:Executive data exposure:** API is generated to get some requests in the web platforms to perform some work, the API generates some more and extra data according to their need and leaks some information in the server platforms [27].

**Preventions Technique:**

To protect the library functions and API functions, developers are required to implement the proper guidance of server authentication and authorization mechanisms. Including some essential security functions that ensure data Routers are strictly limited to necessary implementations.

**2 Insecure Direct Object reference:**

IDOR occurs when applications are exposed in the internal implementations of the object of user interruptions [28]. In the direct web application platforms the IDOR allows the attacker to attack the system software to gain unauthorized access to the actual data of attackers. The identifiers are ready to access the server system integrations[29]. “It helps to conduct a direct interconnections vulnerability that arises when attackers can access or modify objects by manipulating identifiers used in a web application's URLs or parameters. It occurs due to missing access control checks which fail to verify whether a user should be allowed to access specific data. Implementing some essential implementations of a proper access control system to essentially mitigate the risk factor.”

**Attack Mechanism:**

For implementing some consideration of this field are required;

If the URL is considered a user ID(example. com/user?id=123),

If an attacker makes many simple changes in the ID field to access another user's data (example.com/user?id=124).

**Prevention Techniques:**

To provide IDOR vulnerabilities, the application should enforce strict access control and validate some user protection for each request [30].

**3 Security Misconfiguration:**

The security misconfigurations are some incorrect settings in the web application platforms. Servers and the database are the main factors in implementing some validation techniques. This is the phase that can lead to various vulnerabilities such as unnecessary service and the credentials being left unchanged. Regular configuration of management practices can halt the development of best efforts and regular audits of any infrastructure.

**Risk factors**

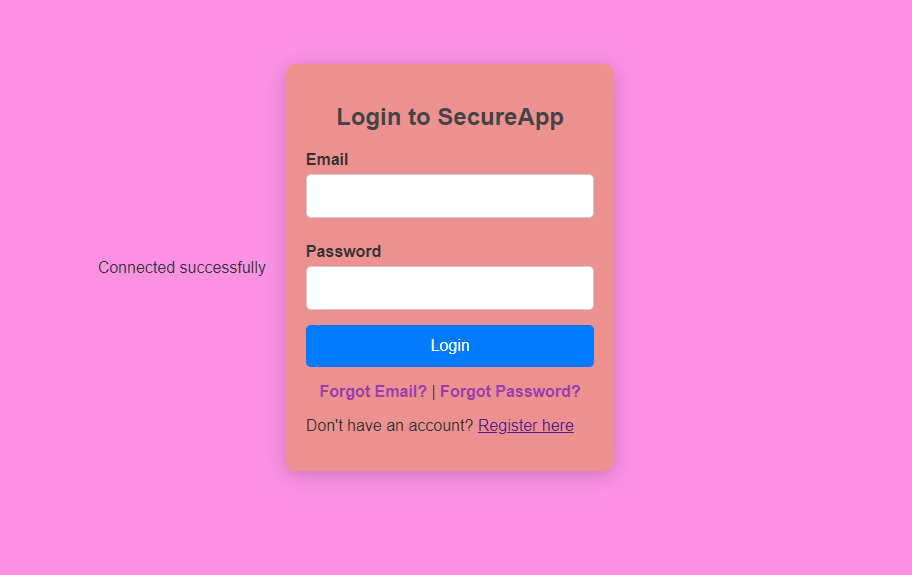
Common misconfigurations include:

1 Default Credentials: Making some changes to manage to change the default user interphase and generate some essential security password inscriptions can leave systems open to attack.

2 Unpatched Software: Running some outdated and older versions of systems software can expose known vulnerabilities.

**Prevention Technique**

In this field, prevention is requested to control the accrual process of the system software in the web application security phases [31]. Regularly write and continuously update security implementations and monitor them because the other users or attackers are attempting to crack the system. Applying some essential security patch files and implementing some hardening practices can help mitigate risk associated with internal and external misconfigurations.



**Figure: Web login pages**

(Source: Self-created)

# Detection and Prevention Techniques

In Real-World Testing DAST simulates real-world attack scenarios allowing organizations to see how their applications will perform under attack [32].

In the case of Identification of Configuration Issues, DAST can uncover misconfigurations and environment-specific vulnerabilities.

**1. “Web Application Firewalls (WAF)**

A barrier between web applications and potential threats is served by WAF. By filtering and monitoring HTTP traffic, WAFs can detect and block malicious requests, offering an additional layer of protection.”

**Features of WAFs**

The Rule-Based Filtering WAF uses attack predefined rules to block common patterns, “such as SQL injection and XSS.”

The Customizable Security Policies of Organizations can create custom rules based on architecture and threat landscape for their specific application [33].

4. Continuous Monitoring and Threat Intelligence

A continuous monitoring approach is leveraging threat intelligence to stay informed about the Organizations should adopt emerging threats and vulnerabilities.

Benefits of Continuous Monitoring

In this case, Proactive Threat Detection is the Continuous monitoring that enables organizations to detect anomalies in real-time allowing for rapid response to potential threats [34]

“The Integration with SIEM Solutions to provide comprehensive visibility into security incidents. Security Information and Event Management (SIEM) systems can aggregate data from various sources, In the case study, Input Validation and Sanitization are the Input fields To mitigate these risks,” they are one of the primary entry points for attacks like SQL Injection and XSS. All user inputs should be validated to ensure they follow expected patterns and data types. Additionally, sanitizing input by escaping special characters ensures that potentially malicious code cannot be executed. Strong authentication mechanisms, such as multi-factor authentication (MFA)can Authentication and Access Control, add layers of security, making it more difficult for attackers to gain unauthorized access., and limiting the potential damage if an account is compromised. “Role-based access control (RBAC) is a common practice where users are assigned specific roles that dictate what they can access or modify, Access control ensures that only authorized users can access specific resources. The database uses secure hashing algorithms like bcrypt or Argon2. Data Encryption is essential for protecting sensitive data both at rest and in transit. For instance, using SSL/TLS (HTTPS) ensures that data transmitted between users and the web application is encrypted, preventing attackers from reading the data. Intercepting and Sensitive information such as passwords should also be encrypted before storing them [35] In the case of Session Management, Secure session management hijacking attacks are crucial for preventing sessions. Implement session timeouts Web applications should use secure cookies (with attributes such as `HttpOnly` and `Secure`), to automatically log out inactive users, and regenerate session IDs after logging in to prevent session fixation attacks. Developers should also regularly update and patch. File can be Outdated software is a common target for attackers, as known vulnerabilities are often exploited. “Web application frameworks, libraries, and plugins should be regularly updated to ensure that security patches are applied. monitor security advisories to stay aware of emerging threats and vulnerabilities.” Testing helps identify and resolve potential vulnerabilities in the important phase of system environments [37]. The Application Security Testing Phase Regular testing of web application vulnerabilities is essential [36]. This can be done through automated tools like vulnerability scanners, penetration testing by security experts, or secure code reviews. Before attackers can exploit them.

# Case Studies

**Case Study 1:** Target Data Breach and Analysis of the Breach

According to Sołtysik-Piorunkiewicz and Krysiak., 2020, Target experienced a massive data breach due to vulnerabilities in its web application and network security. Attackers exploited weaknesses in the company’s payment processing systems, leading to the theft of millions of credit and debit card numbers. This incident underscores the critical importance of securing web applications against both common and emerging threats [38]. This is a part of everyday working life because the application has been considered the total workflow and safety functions of web “application security.” Web application and security software is a part of our daily life. To access Target's network a breach was initiated through compromised vendor credentials allowing attackers to gain access to Insufficient segmentation between networks and access sensitive customer data lack of real-time monitoring allowed the attackers to escalate their privileges. The Target breach highlights the necessity of robust vendor management protection in the network segmentation, and continuous monitoring to detect and respond to threats in real-time.”

**Case Study 2:** Equifax Data Breach and Mitigation Techniques

According to Aziz Al Kabir Elmedany and Sharif., 2023, This case emphasizes the Equifax data breach in 2017 highlights the risks associated with unpatched vulnerabilities. A failure to apply a security update for a known vulnerability in Apache Struts is The breach was primarily due to exposing sensitive information of over 147 million individuals. The increasing prevalence is playing the most critical analysis role in the main frame of server management systems software. They must prioritize patch management Equifax's case underscores the critical importance of maintaining up-to-date software and having a robust vulnerability management program in place. Organizations to prevent the exploitation of known vulnerabilities.

# Future Trends and Research Directions

Future trends may include:

Increased Focus on API Security as more applications integrate with third-party APIs, ensuring their security will be paramount. The adoption of AI and Machine Learning technologies can enhance threat detection and response capabilities, enabling organizations to better anticipate and mitigate risks. In the case of regulatory compliance, as privacy regulations become stricter, organizations will need to adopt security practices that align with these requirements. Integrating security practices into the DevOps pipeline will facilitate a culture of security within development teams, promoting continuous security throughout the software development lifecycle. For research in the future, the analysis of the research could explore the efficacy of various security frameworks, the role of machine learning in threat detection, and the implications of emerging technologies such as blockchain on web application security.

# Conclusion

Web application security is a dynamic and ongoing process that requires vigilance from developers, security experts, and organizations. With the rise of cyber threats and increasingly sophisticated attacks, it is essential to implement security best practices such as input validation, encryption, proper authentication, and regular updates. These measures not only help protect web applications and their users from financial and data losses but also strengthen user trust. Consequently, security has become a critical component of any web development process.

# Reference

1. Zhou, W., Jia, Y., Peng, A., Zhang, Y. and Liu, P., 2018. The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. IEEE Internet of things Journal, 6(2), pp.1606-1616.
2. Alshamrani, A., Myneni, S., Chowdhary, A. and Huang, D., 2019. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. IEEE Communications Surveys & Tutorials, 21(2), pp.1851-1877.
3. Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P. and Sikdar, B., 2019. A survey on IoT security: application areas, security threats, and solution architectures. IEEe Access, 7, pp.82721-82743.
4. Singh, S., Jeong, Y.S. and Park, J.H., 2016. A survey on cloud computing security: Issues, threats, and solutions. Journal of Network and Computer Applications, 75, pp.200-222.
5. Frustaci, M., Pace, P., Aloi, G. and Fortino, G., 2017. Evaluating critical security issues of the IoT world: Present and future challenges. IEEE Internet of things journal, 5(4), pp.2483-2495.
6. Makhdoom, I., Abolhasan, M., Lipman, J., Liu, R.P. and Ni, W., 2018. Anatomy of threats to the internet of things. IEEE communications surveys & tutorials, 21(2), pp.1636-1675.
7. Paul, P.K. and Aithal, P.S., 2019. Database security: An overview and analysis of current trend. International Journal of Management, Technology and Social Sciences (IJMTS), 4(2), pp.53-58.
8. Felderer, M., Büchler, M., Johns, M., Brucker, A.D., Breu, R. and Pretschner, A., 2016. Security testing: A survey. In Advances in Computers (Vol. 101, pp. 1-51). Elsevier.
9. Yevdokymenko, M. and Sokolov, V.Y., 2019. Overview of the Course in “Wireless and Mobile Security”. Educating the Next Generation MSc in Cyber Security, pp.104-119.
10. Tan, T., Wang, B., Tang, Y., Zhou, X. and Han, J., 2019. A Method for Vulnerability Database Quantitative Evaluation. Computers, Materials & Continua, 61(3).
11. Zhao, J., Ji, G., Tian, Y., Chen, Y. and Wang, Z., 2018. Environmental vulnerability assessment for mainland China based on entropy method. Ecological Indicators, 91, pp.410-422.
12. Bialas, A., 2019. Vulnerability assessment of sensor systems. Sensors, 19(11), p.2518.
13. Nguyen, T.T., Bonetti, J., Rogers, K. and Woodroffe, C.D., 2016. Indicator-based assessment of climate-change impacts on coasts: A review of concepts, methodological approaches and vulnerability indices. Ocean & Coastal Management, 123, pp.18-43.
14. Anderson, C.C., Hagenlocher, M., Renaud, F.G., Sebesvari, Z., Cutter, S.L. and Emrich, C.T., 2019. Comparing index-based vulnerability assessments in the Mississippi Delta: Implications of contrasting theories, indicators, and aggregation methodologies. International Journal of Disaster Risk Reduction, 39, p.101128.
15. Zou, D., Wang, S., Xu, S., Li, Z. and Jin, H., 2019. $\mu $ μ VulDeePecker: A Deep Learning-Based System for Multiclass Vulnerability Detection. IEEE Transactions on Dependable and Secure Computing, 18(5), pp.2224-2236.
16. Li, R., Feng, C., Zhang, X. and Tang, C., 2019. A lightweight assisted vulnerability discovery method using deep neural networks. IEEE Access, 7, pp.80079-80092.
17. Machiwal, D., Jha, M.K., Singh, V.P. and Mohan, C., 2018. Assessment and mapping of groundwater vulnerability to pollution: Current status and challenges. Earth-Science Reviews, 185, pp.901-927.
18. Châtelet, E., 2015. Vulnerability and Resilience Assessment of Infrastructures and Networks: Concepts and Methodologies. Risk Management in Life‐Critical Systems, pp.21-40.
19. Zho, Y., Liu, S., Siow, J., Du, X. and Liu, Y., 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. Advances in neural information processing systems, 32.
20. Koroglu, A., Ranasinghe, R., Jiménez, J.A. and Dastgheib, A., 2019. Comparison of coastal vulnerability index applications for Barcelona Province. Ocean & coastal management, 178, p.104799.
21. Shukla, R., Sachdeva, K. and Joshi, P.K., 2018. Demystifying vulnerability assessment of agriculture communities in the Himalayas: a systematic review. Natural Hazards, 91, pp.409-429.
22. Zolanvari, M., Teixeira, M.A., Gupta, L., Khan, K.M. and Jain, R., 2019. Machine learning-based network vulnerability analysis of industrial Internet of Things. IEEE internet of things journal, 6(4), pp.6822-6834.
23. Hassan, M.M., Nipa, S.S., Akter, M., Haque, R., Deepa, F.N., Rahman, M., Siddiqui, M.A. and Sharif, M.H., 2018. Broken authentication and session management vulnerability: a case study of web application. Int. J. Simul. Syst. Sci. Technol, 19(2), pp.1-11.
24. Sarmah, U., Bhattacharyya, D.K. and Kalita, J.K., 2018. A survey of detection methods for XSS attacks. Journal of Network and Computer Applications, 118, pp.113-143.
25. Deepa, G., 2018. Behavior-Based Attack Generation for Detecting Web Application Vulnerabilities (Doctoral dissertation, National Institute of Technology Karnataka, Surathkal).
26. Mack, J., Hu, Y.H.F. and Hoppa, M.A., 2019. A study of existing cross-site scripting detection and prevention techniques using XAMPP and VirtualBox. Virginia Journal of Science, 70(3), p.1.
27. Kumar, A. and Sharma, S., An Analysis of Various Methods to Identify Web Based Applications Vulnerabilities.
28. Cheng, X., Wang, H., Hua, J., Zhang, M., Xu, G., Yi, L. and Sui, Y., 2019, November. Static detection of control-flow-related vulnerabilities using graph embedding. In 2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS) (pp. 41-50). IEEE.
29. Liu, L., De Vel, O., Han, Q.L., Zhang, J. and Xiang, Y., 2018. Detecting and preventing cyber insider threats: A survey. IEEE Communications Surveys & Tutorials, 20(2), pp.1397-1417.
30. Saracino, A., Sgandurra, D., Dini, G. and Martinelli, F., 2016. Madam: Effective and efficient behavior-based android malware detection and prevention. IEEE Transactions on Dependable and Secure Computing, 15(1), pp.83-97.
31. Senyo, P.K., Addae, E. and Boateng, R., 2018. Cloud computing research: A review of research themes, frameworks, methods and future research directions. International Journal of Information Management, 38(1), pp.128-139.
32. Aron, L. and Hanacek, P., 2015, March. Overview of security on mobile devices. In 2015 2nd World Symposium on Web Applications and Networking (WSWAN) (pp. 1-11). IEEE.
33. Hossain, M.M., Fotouhi, M. and Hasan, R., 2015, June. Towards an analysis of security issues, challenges, and open problems in the internet of things. In 2015 ieee world congress on services (pp. 21-28). IEEE.
34. Andrea, I., Chrysostomou, C. and Hadjichristofi, G., 2015, July. Internet of Things: Security vulnerabilities and challenges. In 2015 IEEE symposium on computers and communication (ISCC) (pp. 180-187). IEEE.
35. Abirami, J., Devakunchari, R. and Valliyammai, C., 2015, December. A top web security vulnerability SQL injection attack—Survey. In 2015 Seventh International Conference on Advanced Computing (ICoAC) (pp. 1-9). IEEE.
36. Khan, A., 2016, April. Overview of security in internet of things. In Proceedings of the 3rd International conference on recent trends in engineering science and management, Bundi, Rajasthan, India (Vol. 10).
37. Han, Z., Li, X., Xing, Z., Liu, H. and Feng, Z., 2017, September. Learning to predict severity of software vulnerability using only vulnerability description. In 2017 IEEE International conference on software maintenance and evolution (ICSME) (pp. 125-136). IEEE.
38. Mendoza, A. and Gu, G., 2018, May. Mobile application web api reconnaissance: Web-to-mobile inconsistencies & vulnerabilities. In 2018 IEEE Symposium on Security and Privacy (SP) (pp. 756-769). IEEE.